



POLITECNICO
MILANO 1863

**SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE**

EXECUTIVE SUMMARY OF THE THESIS

FindMyExpert: A Conversational Agent for Academic Collaboration Discovery

LAUREA MAGISTRALE IN COMPUTER SCIENCE - INGEGNERIA INFORMATICA

Author: CESAR ADRIAN CARBAJAL SERRANO

Advisor: PROF. MARK J. CARMAN

Co-advisor: PROF. CRISTIANA BOLCHINI

Academic year: 2025-26

1. Introduction

In international academic institutions, collaboration is a central driver of innovation and research advancement. Within large and renowned institutions such as Politecnico di Milano, the volume of international collaborations is substantial and is reflected in co-authored publications, co-supervised theses, joint projects, co-organised events, and formal agreements with partner institutions. However, the information describing these collaborations is distributed across a broad body of data sources, each providing only a partial and fragmented view of the available relationships. As a consequence, answering questions such as “Who at Politecnico has collaborations with Peruvian universities?”, “Who works on Natural Language Processing?”, or “With whom has a given researcher co-authored publications in Asia?” remains a complex and time-consuming process. The rapid development of Artificial Intelligence (AI) and, in particular, recent advances in Large Language Models (LLMs), have introduced new ways to interact with information. LLMs can interpret user intent expressed in natural language and generate coherent, contextually appropriate

responses, contributing to a growing preference for natural language interfaces over traditional keyword-based search. Despite their capabilities, LLMs exhibit important limitations: their knowledge is bounded by the training data and becomes outdated over time; they are prone to *hallucination*; and they lack direct access to proprietary or domain-specific data. These limitations motivate the integration of LLMs with external retrieval and tool-based mechanisms that can ground model outputs in explicitly retrieved information.

2. Objectives and Research Questions

This thesis proposes *FindMyExpert*, an intelligent conversational system designed to support the exploration of collaboration networks and the identification of researchers working in an area of interest. The system addresses the challenge of fragmented academic information by providing a unified natural language interface over multiple institutional data sources, including publication repositories, thesis archives, and records of international agreements. A key de-

sign goal is the ability to support multi-turn interactions, where users can refine their requests and explore different aspects of the collaboration network in an iterative manner.

The main objective is to design and develop a system enabling users to find academic experts within Politecnico di Milano who are *relevant* in terms of recent international collaborations, and to understand their areas of expertise and collaboration networks. The specific goals are to integrate multiple heterogeneous academic data sources into a unified framework accessible to the end user; to design tools for querying and retrieving relevant information; to enable an LLM to dynamically select and use these tools based on user requests; and to support multi-turn conversations.

Since the proposed work does not assume a fixed system architecture, it follows an iterative design process in which different architectural choices and configurations are explored, implemented, and evaluated, guided by five research questions:

- RQ1.** Can a tool-augmented LLM architecture provide a unified natural language interface, over multiple heterogeneous data sources, that supports expert discovery and collaboration network exploration?
- RQ2.** How do model size, quantization strategy, and reasoning mode influence the trade-off between computational efficiency, reasoning performance, and response quality?
- RQ3.** How do model design choices and inference serving frameworks influence the performance of models in terms of efficiency, response quality, and system behavior?
- RQ4.** How do tool provisioning strategies influence accuracy and system-level behavior in LLM serving systems?
- RQ5.** How does the choice of tool orchestration framework influence response latency and execution reliability in language model systems with external tool integration?

3. Background and Related Work

The proposed system builds on several foundational technologies. Natural Language Process-

ing (NLP) enables the understanding and generation of human language, supporting question answering and dialogue systems. LLMs, built on the transformer architecture [3] and trained through pre-training and fine-tuning phases, provide powerful natural language understanding and generation capabilities, but are limited by outdated knowledge, hallucination, and lack of access to proprietary data.

Retrieval-Augmented Generation (RAG) [1] mitigates these limitations by combining language models with external information retrieval, grounding responses in retrieved documents and reducing hallucination. While effective, standard RAG performs single-pass retrieval and does not support dynamic interaction with structured data sources. Tool-augmented systems [2, 4] extend this paradigm by enabling LLMs to invoke tools, supporting complex multi-step queries. These approaches shift the role of the language model from a passive knowledge source toward an active orchestrator of external capabilities.

In the academic domain, expert discovery systems traditionally rely on keyword-based search, while more advanced approaches exploit semantic representations and collaboration network analysis through knowledge graphs.

4. System Architecture and Implementation

The proposed system follows a modular architecture structured around three main components: (i) a centralized data layer, (ii) a tool-based retrieval framework, and (iii) a language model responsible for orchestration and response generation. As illustrated in Figure 1, a user submits a natural language query; the LLM interprets the user intent and determines which tools are required; the tools interact with the underlying data sources or centralized database to extract the necessary information; the retrieved data is passed back to the model, which produces a final natural language response. This design addresses RQ1, providing a unified natural language interface over heterogeneous academic data sources through a tool-augmented LLM.

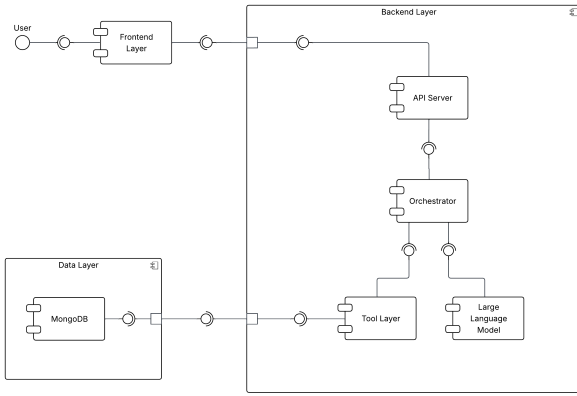


Figure 1: UML component diagram of the implemented system architecture.

4.1. Data integration pipeline

An Extract-Load-Transform (ELT) approach is adopted. Publications are harvested from the IRIS repository and theses from POLITesi, both exposing data through the OAI-PMH protocol, while international agreements are obtained from a structured spreadsheet. Data is first extracted and loaded into a NoSQL database without significant transformations. An enrichment phase retrieves additional metadata for publication records through web scraping. A transformation and normalization phase then standardizes field names, creates derived attributes, and normalizes country and entity names to ensure consistency across sources and improve matching during queries. This architecture decouples data ingestion from data modeling, supporting efficient querying, indexing, and aggregation over heterogeneous datasets.

4.2. Data storage

The system adopts MongoDB, organized into three collections: publications, theses and agreements. The publications and theses collections follow the Dublin Core metadata standard and may include enriched attributes such as affiliations and nationalities, which are critical for identifying international research connections. The agreements collection stores bilateral or multilateral agreements with nested information describing the partner university and associated program. Each collection is self-contained and independent, with no explicit relationships enforced at the database level.

4.3. Tool-based retrieval framework

The LLM does not interact directly with the database. Instead, it selects predefined tools that execute queries over specific collections and return structured results. Tools are organized into four categories: *researcher discovery tools*, which identify POLIMI researchers based on collaboration-related filters; *research output tools*, which retrieve publications and theses linked to a researcher; *collaboration analysis tools*, which extract co-authorship networks, affiliations, and collaboration patterns; and *institutional agreement tools*, which query formal collaboration agreements. Each tool is specialized for a single data source and operates independently.

4.4. LLM integration

The LLM acts as a decision-making component that determines which tool should be invoked and with which parameters, based on the user request and the available context. The system restricts each model decision to a single tool invocation, simplifying the execution logic and reducing ambiguity in tool selection. This two-step process separates information retrieval from response generation, ensuring that all factual content is grounded in explicitly retrieved data rather than model-internal knowledge, thereby reducing hallucinations.

4.5. Implementation

The system was implemented as a modular Python-based backend exposed as a REST API, coordinating data retrieval, tool execution, and interaction with the language model. A web-based frontend provides the conversational interface, with authentication, conversation history management, and a lightweight feedback control on each assistant response. During development, preliminary user testing informed several refinements, including the introduction of data normalization mechanisms to address inconsistencies, and the choice to retrieve only a limited set of the most relevant records for each query, balancing retrieval quality against context-window limitations and GPU memory constraints.

5. Evaluation Methodology

The evaluation assesses the ability to correctly interpret user queries, invoke appropriate tools,

and generate accurate responses. A dedicated test set was manually constructed, consisting of queries designed to trigger the different tools available in the system, including both tool-requiring requests and straightforward factual questions. All experiments were conducted on an NVIDIA T4 GPU with 16 GB of VRAM, limiting the deployable models to the 1B-8B parameter range depending on quantization.

Three complementary metrics are defined:

- **Tool Invocation Accuracy (TIA)** (primary metric): evaluates whether the model calls a tool when necessary and selects the appropriate tool with the correct arguments. A prediction is correct only if the invoked tool matches the expected format, tool, and arguments.
- **Grounded Answer Score (GAS)**: measures response quality with respect to the tool output or previous context, using an LLM-as-a-Judge approach with the xAI Grok-4-1-fast-reasoning model as an automated evaluator.
- **Latency (LAT)**: measures the total time required to generate a response, including model inference and tool execution.

System performance is evaluated as a trade-off between maximizing TIA while jointly optimizing LAT and GAS.

The experimental design varies four main dimensions: language model selection (different families and parameter sizes, with standard and optional reasoning modes, and quantized variants); inference serving frameworks (vLLM and Ollama); tool integration strategies (prompt-based versus schema-based), and tool orchestration frameworks (a custom Python framework versus LangChain).

6. Experimental Results

6.1. Impact of model size and quantization

Results show that TIA is primarily driven by model size and reasoning usage during tool selection. As reported in Table 1, TIA increases steadily from 46.8% for Qwen3-1.7B (reasoning disabled) to 79.7% for Qwen3-4B and 81.4% for Qwen3-8B-FP8 under the same configuration. The introduction of a *mixed reasoning mode*, which applies reasoning only during tool invocation while performing final response genera-

tion without reasoning, proves particularly effective. For Qwen3-8B-FP8, mixed reasoning achieves 91.5% TIA and 98.5% GAS at 25.3 s latency, whereas full reasoning yields 91.2% TIA at 34.1 s, and disabled reasoning drops to 81.4% at 9.6 s. This indicates that inference-time deliberation is most critical for planning and tool selection, while response synthesis requires less deliberation. Quantization has a marginal impact: at the 8B scale, FP8 and AWQ yield nearly identical TIA, while FP8 reaches a higher GAS (98.7% vs. 94.1%) with comparable latency.

Performance across model sizes and quantization levels

Config	Reason.	TIA	GAS	LAT
Qwen3-1.7B	Disabled	46.8%	93.4%	5.1 s
Qwen3-1.7B	Enabled	73.3%	90.8%	16.2 s
Qwen3-4B	Disabled	79.7%	98.2%	10.6 s
Qwen3-4B	Enabled	86.9%	96.3%	38.0 s
Qwen3-4B	Mixed	90.9%	97.7%	28.3 s
Qwen3-8B-AWQ	Disabled	80.9%	94.1%	6.8 s
Qwen3-8B-AWQ	Enabled	91.1%	96.2%	25.6 s
Qwen3-8B-AWQ	Mixed	85.5%	98.6%	19.1 s
Qwen3-8B-FP8	Disabled	81.4%	98.7%	9.6 s
Qwen3-8B-FP8	Enabled	91.2%	98.5%	34.1 s
Qwen3-8B-FP8	Mixed	91.5%	98.5%	25.3 s

Table 1: Impact of model size, quantization, and reasoning mode on system performance

The best-performing configuration is Qwen3-8B-FP8 with mixed reasoning, achieving over 90% TIA and over 98% GAS.

6.2. Impact of model families and serving frameworks

Comparing Qwen3-8B-FP8, Meta-Llama-3.1-8B-Instruct, Phi-3.5-mini-instruct, and Granite-4.1-3b across vLLM and Ollama reveals clear differences, as reported in Table 2.

vLLM generally provides higher performance and lower latency. Qwen3-8B-FP8 achieves consistently high performance in both frameworks, while Llama 3.1 shows a substantial drop under Ollama, and Phi-3.5-mini-instruct falls from 75.3% to 20.6% GAS. Furthermore, a qualitative inspection reveals that Ollama models tend to produce more verbose outputs that violate strict schema requirements, appending natural language commentary to structured tool calls, whereas vLLM-based deployments exhibit more controlled and deterministic output formatting.

Performance across model families and serving frameworks

Model	TIA		GAS		LAT	
	vLLM	Ollama	vLLM	Ollama	vLLM	Ollama
Qwen3-8B-FP8	91.5%	93.6%	98.5%	92.6%	25.3 s	30.4 s
Meta-Llama-3.1-8B-Instruct-FP8	74.0%	49.6%	91.6%	68.1%	9.4 s	12.9 s
Phi-3.5-mini-instruct	40.0%	30.0%	75.3%	20.6%	24.9 s	30.7 s
granite-4.1-3b	52.0%	60.6%	91.7%	82.8%	4.7 s	4.6 s

Table 2: Comparison of TIA, GAS, and LAT across model families and serving frameworks

6.3. Impact of tool provisioning strategies

Two strategies are compared, as reported in Table 3: *prompt-based* tool specification, where tool descriptions are embedded in the system prompt, and *schema-based* tool specification, where tools are passed through the native framework interface with a dedicated structured output field.

Performance across tool provisioning strategies (Qwen3-8B-FP8)

Reason.	Provisioning	TIA	GAS	LAT
Disabled	System prompt	81.4%	98.7%	9.6 s
Mixed	System prompt	91.5%	98.5%	25.3 s
Disabled	Tool schema	87.4%	90.7%	10.7 s
Mixed	Tool schema	88.8%	91.8%	27.5 s

Table 3: Impact of tool provisioning strategy and reasoning mode on system performance

Under disabled reasoning, schema-based provisioning improves TIA from 81.4% to 87.4% with comparable latency, but reduces GAS from 98.7% to 90.7%. Prompt-based provisioning benefits more from reasoning: mixed reasoning raises TIA to 91.5% while keeping GAS stable at 98.5%. This reveals a strong interaction effect: schema-based definitions improve baseline robustness under low reasoning but become more sensitive under increased reasoning complexity. The optimal provisioning strategy is therefore context-dependent: prompt-based provisioning with reasoning augmentation is preferable when accuracy is the priority, while schema-based provisioning without reasoning suits latency-sensitive deployments.

6.4. Impact of tool orchestration frameworks

Our custom Python-based framework is compared with LangChain, as reported in Table 4. Under comparable conditions (Qwen3-8B-FP8, reasoning disabled, tool schema), the custom framework achieves 87.4% TIA, 90.7% GAS, and 10.7s latency, while LangChain reaches 84.5% TIA, 88.6% GAS, and 12.3s latency. With mixed reasoning enabled in the custom framework, the gap widens to 91.5% TIA and 98.5% GAS.

Two factors explain this difference. First, LangChain’s reasoning configuration is constrained to a global binary setting, preventing the mixed reasoning strategy that proved beneficial; enabling reasoning globally incurs approximately 55s latency, making it impractical. Second, the custom framework implements a fallback parsing layer that recovers tool invocations from both structured output fields and general response content, whereas LangChain relies primarily on native structured outputs. These results suggest that higher-level orchestration frameworks provide usability benefits but may introduce constraints in reasoning control and output robustness that negatively affect performance in tool-intensive settings.

7. Conclusions

This thesis proposes *FindMyExpert*, an intelligent conversational system that assists faculty at Politecnico di Milano in exploring collaboration networks and identifying relevant experts. The work delivers a data integration pipeline that collects, transforms, and normalizes information from heterogeneous institutional sources into a centralized MongoDB database; a tool-based retrieval framework that enables structured access to academic data through dedicated tools; and

Performance across orchestration frameworks

Model	Reason.	Framework	Provisioning	TIA	GAS	LAT
Qwen3-8B-FP8	Mixed	Custom	System prompt	91.5%	98.5%	25.3 s
Qwen3-8B-FP8	Disabled	Custom	Tool schema	87.4%	90.7%	10.7 s
Qwen3-8B-FP8	Disabled	LangChain	Tool schema	84.5%	88.6%	12.3 s

Table 4: Impact of orchestration framework on system performance

a system architecture in which an LLM acts as an orchestrator, dynamically selecting tools and synthesizing retrieved information into coherent natural language responses. A comprehensive evaluation analyzed the impact of model size, quantization, reasoning mode, serving framework, tool provisioning strategy, and orchestration approach on system performance.

The evaluation provides a positive answer to RQ1: when appropriately configured, the proposed tool-augmented LLM architecture delivers a unified natural language interface over consolidated heterogeneous academic data sources, supporting expert discovery and collaboration network exploration. The best-performing configuration, Qwen3-8B-FP8 under vLLM with mixed reasoning, achieves 91.5% TIA, 98.5% GAS, and 25.3 s latency.

The findings yield five main insights: (i) tool calling scales with model size; (ii) mixed reasoning provides the best accuracy-efficiency trade-off; (iii) quantization has marginal impact on accuracy; (iv) the serving framework influences output behavior and not only latency, with vLLM better suited for structured outputs; and (v) tool provisioning interacts non-linearly with reasoning, making the optimal strategy context-dependent.

Finally, our custom orchestration framework outperforms an established library such as LangChain in tool-intensive settings, due to finer-grained reasoning control and more resilient output parsing.

The evaluation was constrained by the limited available GPU resources, preventing the testing of larger models that may exhibit qualitatively different behavior. Future directions include exploring larger reasoning-capable model families with more powerful hardware, reducing reliance on predefined tools by exposing the database schema directly to the language model, complementing the LLM-as-a-Judge approach with

human evaluations, and leveraging deployment-time user feedback for fine-tuning. The data integration pipeline can accommodate additional sources with minimal changes, such as joint research projects or co-organised events, to further enrich system coverage.

Overall, *FindMyExpert* demonstrates strong tool-calling capabilities when appropriately configured, offering a solid foundation for further development and deployment in real-world academic settings.

References

- [1] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.
- [2] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc., 2023.
- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [4] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2023.